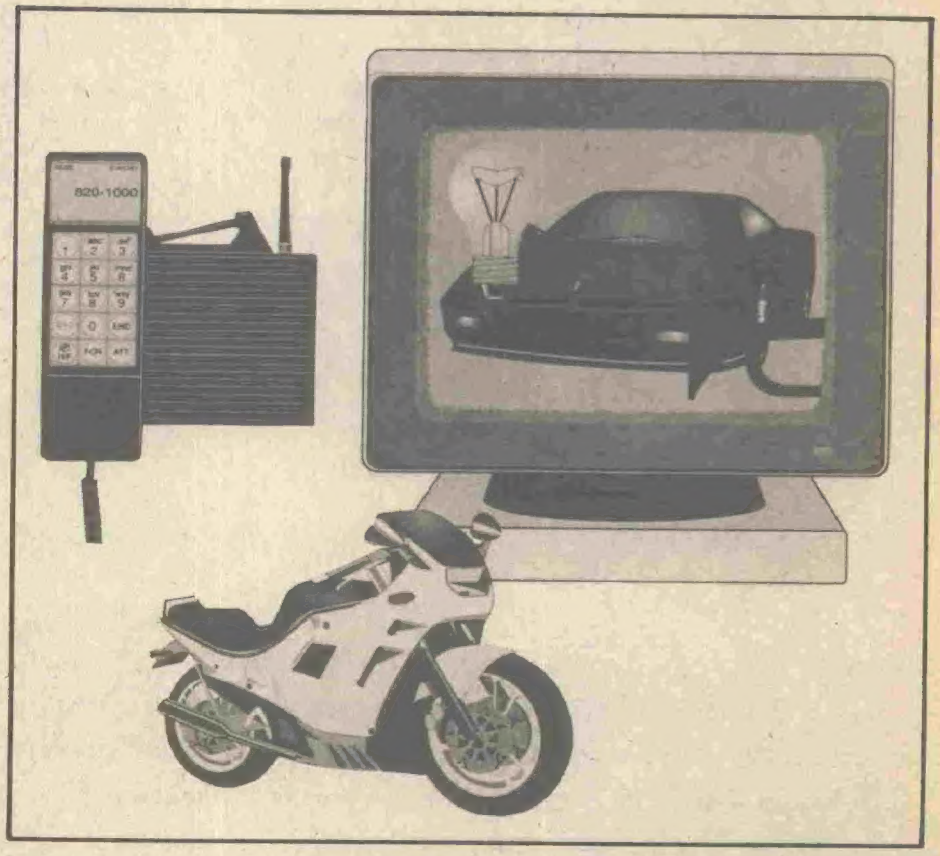
ซอฟต์แวร์
อนันต์ กิตติบรรดุ

วินโดวส์ โปรแกรมมิง : MENU

ศึกษาวิธีการติดต่อกับผู้ใช้ของโปรแกรมบนวินโดวส์ พร้อมกับวิธีการสร้างเมนู และการปรับปรุงเมนู เพื่อใช้ในแอพพลิเคชันต่อไป



วินโดวส์โปรแกรมมิงในตอนนี้เราจะมาพูดกันในเรื่องของเมนูกันนะครับ ซึ่งเป็นส่วนที่ใช้เพื่อการติดต่อกับผู้ใช้ได้เป็นอย่างดีทำความเข้าใจได้ง่ายและสะดวกในการจดจำมากกว่าการที่ผู้ใช้ต้องมาจดจำคำสั่งต่าง ๆ ต้องมาคอยเปิดคู่มือการใช้งานประกอบไปด้วย สร้างความยุ่งยากและน่าเบื่อ ด้วยเหตุนี้แหละครับที่ GUI ทุก ๆ ตัวจะต้องผนวกเรื่องเมนูไว้ให้แอพพลิเคชันต่าง ๆ เรียกใช้ไว้ด้วย

วิธีสร้างเมนูอย่างง่าย

จริง ๆ แล้วการสร้างเมนูบนวินโดวส์นั้นสะดวกและง่ายดายมาก แต่จะมีที่ยากขึ้นมาหน่อยก็เป็นพวก Floating Pop-up Menu ผมขอเรียกง่าย ๆ ว่าเมนูลอยตัว, พวกบิตแม็พเมนู (BITMAP MENU) เป็นต้น ซึ่งเดี๋ยวเราค่อยมาว่ากันต่อไป ตอนนี้เอาอย่างง่าย ๆ ก่อน และส่วนใหญ่ก็ใช้วิธีนี้ทั้งนั้น ขั้นตอนคือ..

1. สร้างไฟล์รีซอร์สสคริปต์ (.RC) ที่บอกชื่อและลักษณะเมนูเอาไว้

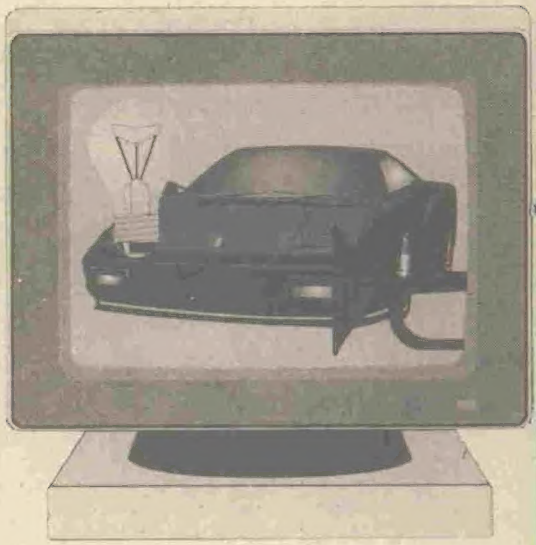
2. ในส่วนซอร์สโค้ดโปรแกรม (.C) ซึ่งจะนำเมนูจากรีซอร์สใน (1) มาโหลดสู่แอพพลิเคชัน โดยสามารถเรียกใช้ได้ 2 ฟังก์ชันดังนี้

2.1 RegisterClass(..) เมื่อตอนที่เริ่มทำการลงทะเบียนสร้างคลาสใหม่ เพื่อให้คลาสนี้ไปใช้สร้างวินโดว์ใหม่ ซึ่งเราบอกชื่อเมนูในไฟล์รีซอร์สได้จากฟิลด์ lpszMenuName ที่มาจากเรคคอร์ดของ WNDCLASS (วิธีนี้เป็นวิธีที่ง่าย สะดวกและนิยมใช้กันมากที่สุด)

2.2 CreateWindow(..) ซึ่งลักษณะการส่งจะแตกต่างกันออกไปจาก 2.1 ตรงที่ว่าเราต้องส่งค่าแฮนเดิลของเมนูไป ค่านี้จะได้จากการเรียก LoadMenu( ) เพื่อที่จะนำชื่อเมนูจากรีซอร์สโหลดเข้าสู่เมมโมรี จึงจะได้ค่าแฮนเดิลเป็นค่าอ้างอิงส่งให้กับ Create Window( ) ต่อไป

```
(1) ExampleMenu MENU
    BEGIN
    (2) MENUITEM "&One", ID_1
    (3) POPUP "&Two"
        BEGIN
        (4) MENUITEM "Two-&one", ID_2_1, CHECKED
            MENUITEM "Two-&two", ID_2_2
        (5) MENUITEM SEPARATOR
            MENUITEM "Two-t&hree", ID_2_3
        (6) POPUP "Two-four"
            BEGIN
                MENUITEM "Two-Four-&One", ID_2_4_1
                MENUITEM "Two-Four-&two", ID_2_4_2
            END
            MENUITEM "Two-&five", ID_2_5
            END
        END
        MENUITEM "T&hree", ID_3
    (7) MENUITEM "&Four", ID_4, GRAYED
        MENUITEM "Fi&ve", ID_5
    (8) MENUITEM "Si&x", ID_6, MENUBREAK
        MENUITEM "Se&ven", ID_7
        MENUITEM "&Eight", ID_8
    END
```

รูป A
ตัวอย่างการสร้างเมนู
ในรีซอร์สไฟล์



(1) เป็นการบอกให้ตัวรีซอร์สคอมพายเลอร์ทราบว่า ExampleMenu เป็นชื่อของเมนู ดังจะเห็นได้จาก MENU ที่เป็นคีย์เวิร์ดของเมนูที่ตามมา ซึ่งชื่อนี้จะเป็นชื่อที่ใช้อ้างอิงให้วินโดวส์ทราบว่าจะใช้เป็นเมนู

(2) MENUITEM เป็นคีย์เวิร์ดที่บอกว่าให้นำสตริง "One" ไปใส่ไว้ในรายการของเมนูด้วย กล่าวคือเมื่อผู้ใช้เลื่อนเม้าส์มากดบริเวณนี้ หรือกด ALT-O (เนื่องด้วยชื่อนี้มีเครื่องหมาย '&' (อ่านว่า แอมเพอแซนด์) นำหน้าตัว 'O' จึงทำให้วินโดวส์ทราบว่าเป็น Hot-Key ของรายการนี้ และวินโดวส์ก็จะแสดงตัวอักษรที่ตามมาแบบขีดเส้นใต้) แล้ววินโดวส์จะแสดงเป็นแถบทึบ และหลังจากที่ผู้ใช้เลือกรายการนี้วินโดวส์จะทำการส่งเมสเสจ WM_COMMAND ไปให้ส่วนวินโดว์ฟังก์ชัน ซึ่งเราจะทราบว่าเป็นเมนูของรายการนี้ได้จากค่า wParam (พารามิเตอร์ตัวที่ 3 ของวินโดว์ฟังก์ชัน) จะเป็น ID_1 (โดยวิธีการใช้ switch (wParam) และ case ID_1) แล้วจะทำงานอะไรต่อไป ก็เป็นหน้าที่ของแอพพลิเคชันนั้น (ดูรูปที่ 3)

(3) POPUP เป็นคีย์เวิร์ดที่ต่างจาก (2) ตรงที่ว่า เมื่อผู้ใช้เลือกรายการนี้จะมีรายการย่อยตามมาอีก (ดูรูปที่ 2) โดยวินโดวส์จะนำรายการตั้งแต่ Two-one ไปจนถึง Two-five ขึ้นมาแสดง และข้อพึงสังเกตคือ ถ้าเป็นรายการเมนูแบบ POPUP แล้วไม่ต้องมีค่า ID อะไรเลยเพราะมันยังไม่สิ้นสุดการเลือกรายการ (ดูรูปที่ 2)

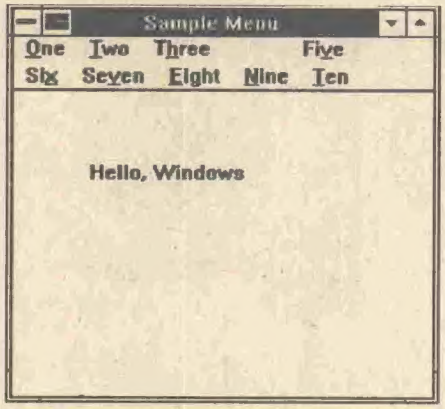
(4) จะมีคีย์เวิร์ด CHECKED ซึ่งจะทำให้วินโดวส์แสดงเครื่องหมาย check mark (เครื่องหมายขีดถูก) นำหน้ารายการนี้ สำหรับการทำเช่นนี้ก็ใช้เพื่อให้ผู้ใช้จะได้ทราบว่าตอนนี้เขาเลือกอะไรอยู่ เช่นในขณะนั้น Sound เปิดหรือปิดอยู่ เป็นต้น (ดูรูปที่ 2)

(5) จะมีคีย์เวิร์ด SEPARATOR ทำการขีดเส้นเพื่อแยกรายการนี้ออกจากรายการถัดไป

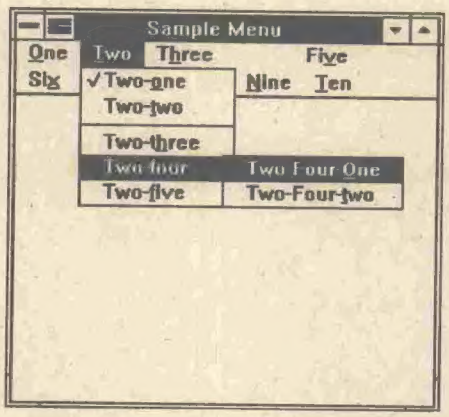
(6) เป็นเมนูแบบ POPUP อีกครั้งหนึ่ง ซึ่งเมนูลักษณะเช่นนี้เรียกว่า "Cascade Menu" (ดังรูปที่ 2)

(7) จะมีคีย์เวิร์ด GRAYED ซึ่งจะบอกให้วินโดวส์และผู้ใช้ทราบว่ารายการนี้ยังไม่ยินยอมให้ใช้ (ถูกดิสเอเบิ้ล) โดยวินโดวส์จะแสดงสตริง "Four" เป็นแบบตัวจาง (gray text)

(8) จะมีคีย์เวิร์ด MENUBREAK เพื่อให้วินโดวส์ทำการแยกรายการนี้ออกจากรายการถัดไป (แตกต่างจาก (5) ตรงที่ (5) จะใช้ภายในรายการ POPUP) เราจะใช้วิธีนี้เมื่อเห็นว่ารายการเมนูเหล่านั้นกว้างเกินไป



รูปที่ 1 แสดงผลลัพธ์จากการสร้างเมนู



รูปที่ 2 แสดงรายการเมนูแบบป๊อปอัพ

```
WndClass.lpszMenuName = "ExampleMenu";
ซึ่งเป็นการให้ค่าสตริง lpszMenuName ชี้ไปที่ชื่อ ExampleMenu
และหลังจากที่ให้ค่าทุกฟิลด์ใน WndClass (จากเรคคอร์ด WNDCLASS) แล้ว เรียก
RegisterClass(&WndClass);
ก็เป็นอันได้เมนูมาอยู่บนแอพพลิเคชันแล้ว
```

รูป B วิธีลงทะเบียนคลาส

จากลักษณะของรีซอร์สดังกล่าวจะได้เมนูเป็นดังรูปที่ 1 และ 2

ที่นี้มาดูวิธีการโหลดเมนูจากไฟล์รีซอร์สมาใส่ในแอพพลิเคชัน เช่นจะนำเมนูที่ชื่อ "EaxmpleMenu" จากตัวอย่างด้านบนมาใช้ได้ดังนี้

วิธีที่ 1 โดยวิธีลงทะเบียนสร้างคลาส ดังแสดงในรูป B

วิธีที่ 2 โดยวิธีสร้างวินโดว์ดังแสดงในรูป C

สำหรับค่า ID ต่าง ๆ ก็เป็นค่าจำนวนเต็มบวกที่เราจะดีฟายน์ไว้ในเฮดเดอร์ไฟล์ เช่น

```
#define ID_1      100
#define ID_2_1    210
  "
#define ID_8      800
```

การเปลี่ยนแปลงเมนูหรือรายการในเมนู

จากตัวอย่างด้านบนที่ผมกล่าวไปนั้นเป็นเพียงการสร้างเมนูเท่านั้น แต่การทำงานของเมนูขณะที่แอพพลิเคชันยังรันอยู่นั้นยังสามารถที่จะเปลี่ยนแปลงได้ด้วยเช่น

1. ต้องมีการอีนาเบิ้ล/ดิสเอเบิ้ลบางรายการ
2. เปลี่ยนแปลงเครื่องหมายเช็คมาร์คของรายการย่อย
3. เพิ่มหรือลบบางรายการออกไป
4. เปลี่ยนใหม่ทั้งเมนูเลย

การอีนาเบิ้ล/ดิสเอเบิ้ลรายการ

เมื่อตอนต้นผมได้พูดถึงคีย์เวิร์ด GRAYED จากรายการที่ 4 ไปแล้วว่าเพื่อแสดงว่ารายการนั้นได้ถูกดิสเอเบิ้ลไม่สามารถที่จะเลือกได้ ทีนี้ถ้าถึงเวลาให้รายการนี้เลือกได้แล้วละก็ เราจะเปลี่ยนมันได้โดยเรียก..

```
EnableMenuItem (hMenu, ID_4, MF_ENABLED);
```

ค่า hMenu ได้มาจากการเรียก hMenu=LoadMenu (..) เมื่อตอนเริ่มรันแอพพลิเคชัน (เป็นการสร้างเมนูจากรีซอร์สไฟล์ขึ้นมา) หรืออาจได้จากการเรียก hMenu=GetMenu (hWnd); (กรณีที่มีเมนูอยู่แล้ว (จากการบอกชื่อเมนูเมื่อตอนลงทะเบียนสร้างคลาส)

และหากจะกลับสู่การดิสเอเบิ้ลอีกครั้งโดยการเรียก...

```
(1) EnableMenuItem (hMenu, ID_4, MF_DISABLED);
หรือ
(2) EnableMenuItem (hMenu, ID_4, MF_GRAYED);
```

สำหรับ (1) และ (2) จะแตกต่างกันตรงที่ว่า (1) จะแสดงข้อความของรายการตามปกติ แต่ (2) จะแสดงแบบตัวจาง (gray text) ซึ่งเป็นวิธีที่นิยมใช้กันมากกว่า

การเปลี่ยนแปลงเครื่องหมายเช็คมาร์คของรายการย่อย

จากรายการย่อยที่ 1 ของรายการที่ 2 เมื่อตอนต้น จะมีคีย์เวิร์ด CHECKED อยู่เพื่อเป็นการทำเครื่องหมายถูกไว้ด้านหน้าของรายการนี้ให้ผู้ใช้ทราบว่าตอนนี้อยู่ในรายการอะไร และหากเราต้องการเปลี่ยนไปรายการอื่น เช่นเป็นรายการย่อยที่ 2 ของรายการที่ 2 แทน จะต้องทำ 2 ขั้นตอนด้วยกันคือ

```
HWND hWnd;
HMENU hMenu;
  ..
  ..
hMenu = LoadMenu (hInstance, "ExampleMenu");
hWnd  = CreateWindow (
        "Example-WindowClass", /*ชื่อ class*/
        "Example-Title",       /*ชื่อ title bar*/
        WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT,  CW_USEDEFAULT,
    CW_USEDEFAULT,  CW_USEDEFAULT,
        NULL,    /*Parent window ในที่นี้ไม่มีจึงใช้ NULL*/
        hMenu,   /*ชื่อเมนูที่ได้จากการโหลด*/
        hInstance, NULL);
```

รูป C
วิธีสร้างวินโดว์

```
HMENU hOldMenu, hNewMenu;
  ..
  ..
hOldMenu = GetMenu (hWnd); // อ่านค่าแฮนเดิลของเมนูที่อยู่บนแอพพลิเคชันตอนนั้น
hNewMenu = LoadMenu (hInstance, "SkillMenu");
Setmenu (hWnd, hNewMenu); // เป็นฟังก์ชันที่จะนำเอาแฮนเดิล hNewMenu ไปเป็นเมนูใหม่บนแอพพลิเคชัน
```

รูป D

1. CheckMenuItem (hMenu, ID_2_1, MF_UNCHECKED);
2. CheckMenuItem (hMenu, ID_2_2, MF_CHECKED);

ใน (1) จะทำให้เครื่องหมายถูกหายไป และ (2) นำเครื่องหมายถูกไว้หน้ารายการ

การเพิ่มรายการเข้า

การเพิ่มรายการเข้าไปทำได้ 2 วิธี

1. เพิ่มแบบต่อท้ายรายการ (ใช้ง่ายกว่า) เช่นเราจะเพิ่มรายการที่ 9 เข้าไว้ในเมนูด้วย ได้โดย..

AppendMenu (hMenu, MF_ENABLED, ID_9, "&Nine"

2. เพิ่มแบบแทรกรายการ เช่นเราจะแทรกรายการที่ 0 ไว้ก่อนหน้ารายการที่ 1 ได้โดย...

InsertMenu (hMenu, ID_1, MF_BYCOMMAND | MF_GRAYED, ID_0, "Zero");

เป็นการแทรกให้รายการเป็นดิสเอเบิ้ลแบบ GRAYED ส่วน MF_BYCOMMAND เป็นการบอกให้ทราบว่าค่า ID_1 ที่ส่งไปเป็นของเมนู แต่ถ้าเป็น..

InsertMenu (hMenu, 0, MF_BYPOSITION | MF_GRAYED, ID_0, "Zero")

จะเห็นว่าการส่ง MF_BYPOSITION ไปก็เพื่อบอกให้วินโดวส์รู้ว่าค่า 0 นี้ เป็นค่าของตำแหน่ง (เริ่มจากรายการแรกสุดจะเป็น 0 รายการถัดไปเป็น 1 2 3 ..ตามลำดับ)

การลบรายการออก

การลบรายการออกทำได้โดย เรียกฟังก์ชัน DeleteMenu( ) เช่นถ้าจะเอารายการที่ 2 ออกไป โดย..

DeleteMenu (hMenu, ID_2, MF_BYCOMMAND);

หรือ

DeleteMenu (hMenu, 1, MF_BYPOSITION);

การเปลี่ยนเมนูใหม่

เราสามารถที่จะเปลี่ยนจากเมนูหนึ่งไปสู่อีกเมนูหนึ่งได้ เช่นต้องการเปลี่ยน ExampleMenu ไปเป็น SkillMenu ได้ดังแสดงในรูป D

การผนวกฮอตคีย์ไว้กับเมนู

สำหรับบนวินโดวส์แล้วเรียกฮอตคีย์กันว่า "Accelerator Key" ลักษณะคีย์เช่นนี้มีความพิเศษที่ว่าต้องการให้การเข้าไปในรายการต่าง ๆ ในเมนูเป็นไปด้วยความรวดเร็ว เช่น การกดคีย์ ALT+X หมายถึงออกจากแอพพลิเคชัน (เร็วกว่าการเคลื่อนเม้าส์ไปที่เมนูแล้วเลือกรายการ Exit)

วิธีการผนวกแอกซิลีเรเตอร์คีย์ไว้ในแอพพลิเคชันมีขั้นตอนดังนี้

1. บอกชื่อรายการเมนูในไฟล์รีซอร์สไว้ให้ผู้ใช้ทราบว่าคีย์ดังกล่าวคืออะไร เช่นถ้าต้องการให้รายการที่ 1 ของเมนูสามารถใช้คีย์ ALT+F1 ได้ด้วยเราต้องแก้ไฟล์รีซอร์สใหม่เป็น

```
MENUITEM "&One \tAlt+F1", ID_1
// เครื่องหมาย '/t' คือตัว tab
```

2. เพิ่มตารางแอกซีลีเรเตอร์ไว้ในรีซอร์ส โดยใช้คีย์เวิร์ด ACCELERATORS ดังรูปที่ 7 VK_F1 เป็น virtual key (เป็นคีย์ที่วินโดวส์ทำการแปลงรหัสแล้ว เพื่อให้สามารถซัพพอร์ตได้กับคีย์บอร์ดหลายแบบ) ซึ่งเราต้องบอกให้ตัวรีซอร์สคอมพายเลอร์ทราบว่าเป็นเวอร์ชวลคีย์ด้วยการใส่คีย์เวิร์ด VIRTKEY ไว้ด้วย และที่ตามมาเป็น ALT (ถ้าเป็นตัวชิฟตัวอื่น ๆ ใช้ SHIFT, CONTROL เป็นต้น แต่ถ้าไม่ต้องการชิฟคีย์เหล่านี้ก็ไม่ต้องใส่ลงไป)

3. เพิ่มโค้ดการโหลดคีย์แอกซีลีเรเตอร์จากตารางไว้ในแอพพลิเคชันในซอร์สโปรแกรมหลักเมื่อตอนที่ได้รับเมสเสจ WM_CREATE ดังแสดงในรูป E

4. เปลี่ยนเมสเสจลูปให้วินโดวส์รับรู้แอกซีลีเรเตอร์คีย์เหล่านี้ ดังรูปที่ 4 จากรูปอธิบายได้ดังนี้

(1) เป็นการตรวจว่าเมสเสจที่ได้มานั้นเป็นแอกซีลีเรเตอร์คีย์ของ hHotKeyExampleMenu หรือไม่

ถ้าใช่แล้วเมสเสจนั้นจะถูกวินโดวส์แปลงให้เป็นเมสเสจ WM_COMMAND โดยมีค่า wParam เป็น ID_1 เพื่อส่งให้กับวินโดว์ฟังก์ชัน hWnd ไปทำงานต่อไป

(2) เมสเสจนั้นไม่ใช่แอกซีลีเรเตอร์คีย์ เมสเสจนั้นก็จะถูกส่งไปทำงานปกติด้วยฟังก์ชัน TraslateMessage และ DispatchMessage

```
case WM_COMMAND:
    switch (wParam) {
        case ID_1:
            /* เรียกฟังก์ชันที่จัดการเกี่ยวกับรายการที่ 1 */
        break;
        case ID_2_1:
            /* เรียกฟังก์ชันที่จัดการเกี่ยวกับรายการย่อยที่ 1 ของรายการที่ 2 */
        break;
            ..
            ..
        case ID_8:
            /* เรียกฟังก์ชันที่จัดการเกี่ยวกับรายการที่ 8 */
        break;
    }
```

รูปที่ 3 แสดงตัวอย่างการทำงานเพื่อรับข้อมูลจากวินโดวส์หลังจากที่ผู้ใช้เลือกรายการแล้ว

```
while (GetMessage(&msg, NULL, NULL, NULL) {
    (1) if (!TraslateAccelerators(hWnd, hHotKeyExampleMenu, &msg)) {
    (2)     TranslateMessage (&msg);
            DispatchMessage (&msg);
        }
}
```

รูปที่ 4 แสดงส่วนเมสเสจลูปเมื่อมีแอกซีลีเรเตอร์คีย์

```
HANDLE hInst;
HANDLE hHotKeyExampleMenu;
    ..
    ..
case WM_CREATE:
    hHotKeyExampleMenu = LoadAccelerators (hInst, "HotKeyExampleMenu");
    ..
    break;
```

การสร้างเมนูแบบลอยตัว

เมนูแบบลอยตัว (Pop-up Menu) เป็นเมนูที่มีความพิเศษที่ว่ามันจะไม่อยู่

```
case WM_RBUTTONDOWN: {
      POINT pt;
      HMENU hFloatPopupMenu;
    (1) pt = MAKEPOINT (lParam); // from long int to be point
    (2) hFloatPopupMenu = CreatePopupMenu ( );
    (3) AppendMenu (hFloatPopupMenu, MF_ENABLED, ID_CHOICE_A, "Choice_A");
        AppendMenu (hFloatPopupMenu, MF_ENABLED, ID_CHOICE_B, "Choice_B");
        AppendMenu (hFloatPopupMenu, MF_ENABLED, ID_CHOICE_C, "Choice_C");
        AppendMenu (hFloatPopupMenu, MF_ENABLED, ID_CHOICE_D, "Choice_D");
    (4) ClientToScreen (hWnd, (LPPOINT) &pt); // from window's point to screen's point
    (5) TrackPopupMenu (hFloatPopupMenu, NULL, pt.x, pt.y, NULL, hWnd, NULL);
    (6) Destroy Menu (hFloatPopupMenu);
        break;
}
```

รูปที่ 5 แสดงวิธีการสร้างเมนูลอยตัว

ถาวรบนวินโดว์ แต่จะป๊อปอัพขึ้นมาในตำแหน่งของพอยน์เตอร์ตอนนั้นแล้วแต่ว่าแอพพลิเคชันตัวนั้นจะกำหนด เช่น จะแสดงเมื่อมีการกดเม้าส์ปุ่มซ้าย กลาง หรือขวา หรือกดคีย์บอร์ด เป็นต้น

การสร้างเมนูแบบลอยตัว ก็มีลักษณะคล้ายกับการสร้างเมนูที่อยู่บนวินโดว์เช่นกัน แต่ฟังก์ชันที่เรียกนั้นต่างกันเล็กน้อย คือเรียกจาก CreatePopupMenu และ TrackPopupMenu คู่กัน ดังเช่นตัวอย่างในรูปที่ 5 เป็นการสร้างเมนูแบบลอยตัวเมื่อผู้ใช้กดเม้าส์ปุ่มขวาลงบนวินโดว์ และจะได้เอาต์พุตเป็นดังรูปที่ 6

จากตัวอย่างรูปที่ 5

(1) ฟังก์ชัน MAKEPOINT จะทำการเปลี่ยนตำแหน่งที่วินโดวส์ส่งมาเป็นค่า long integer (ค่าเวิร์ดต่ำเป็น x และเวิร์ดสูงเป็น y) ให้กลายเป็นโครงสร้างแบบ point

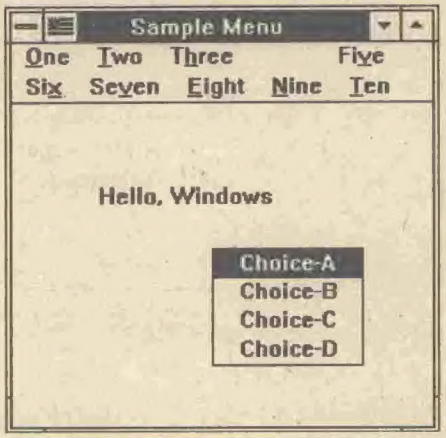
(2) สร้างป๊อปอัพเมนูขึ้นมา

(3) เพิ่มเมนูแบบต่อท้ายเมนูใน (2)

(4) เป็นฟังก์ชันที่ทำการเปลี่ยนตำแหน่งจากที่อ้างอิงกับวินโดว์นั้น (logical point) เป็นอ้างอิงกับสกรีน (screen point)

(5) ทำการแสดงเมนูที่ได้รับการสร้างมาในตำแหน่งที่ (4)

(6) ทำการทำลายเมนูที่สร้างขึ้นเพื่อให้นำเมมโมรีส่วนนี้ไปใช้ต่อไปได้

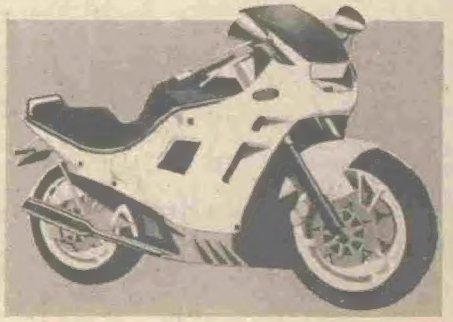


รูปที่ 6 แสดงหน้าเอาต์พุตที่สร้างตามข้อ 5

```
HotKeyExampleMenu ACCELERATORS
BEGIN
    VK_F1, ID_1, VIRTKEY, ALT
END
```

รูปที่ 3 Accelerator-table

ส่งท้าย

ผมก็ขอจบเรื่องของเมนูเอาไว้เพียงเท่านี้นะครับ ซึ่งเนื้อหาก็คงจะครอบคลุมเรื่องการสร้างเมนูทุกวิธีแล้ว คงเหลือเพียงเรื่องบิตแม็พเมนู ที่ผมขอยกยอดไว้พูดกันวันหลัง เพราะยังมีเรื่องน่าสนใจอื่น ๆ อีกมากที่จะทยอยตามกันมาให้ท่านผู้อ่านได้รับรู้กัน และเช่นเคยของคอลัมน์นี้ก็ตัวอย่างโปรแกรมที่จะขาดเสียมิได้ เพื่อประกอบความเข้าใจได้รวดเร็วขึ้น สำหรับตัวอย่างเมนูในตอนนี้ผมก็นำเอาตัวอย่าง Hellowin.exe มาปัดฝุ่นใหม่ก็ลองคอมพายล์กันดูนะครับ สำหรับฉบับนี้ก็ลากันเพียงเท่านี้ครับ สวัสดี.

ส่วนตัวอย่างโปรแกรมได้เตรียมไว้ให้ในซอฟต์เมล์ฉบับนี้แล้วครับ